Investigating performance of the XAMG library for solving linear systems with multiple right-hand sides

Boris Krasnopolsky, **Alexey V. Medvedev**
Institute of Mechanics,
Lomonosov Moscow State University

НАУЧНАЯ КОНФЕРЕНЦИЯ
Суперкомпьютерные дни в России

The presented work is supported by RSF grant No. 18-71-10075.
The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

XAMG project outline

XAMG:

- Is a C++ library to solver large-scale sparse linear systems (SLAEs)
- with multiple right-hand sides (RHS)

  → iterative methods: BiCGStab + CAMG + smoothers

  → Multicore + MPI + (GPU: WIP)

- RSF grant No. 18-71-10075

B.Krasnopolsky, A.Medvedev
«XAMG: A library for solving linear systems with multiple right-hand side vectors»

- git: https://gitlab.com/xamg

  License: dual licensed: GPL or commercial

Multiple RHS

$$|A| \cdot x_1 = b_1$$

$$|A| \cdot x_2 = b_2$$

$$\dots$$

$$|A| \cdot x_n = b_n$$

$$|A| \cdot \{ x_1, x_2, \dots, x_n\} = \{b_1, b_2, \dots, b_n\}$$

Multiple RHS

Solution with multiple RHSs

vs.

Multiple runs with single RHS

Speedup level?

Multiple RHS

Solution with multiple RHSs

vs.

Multiple runs with single RHS

Speedup level?

B.Krasnopolsky

«Revisiting performance of BiCGStab methods for solving systems with multiple right-hand sides»

Predicted speedup: **~1.5x … 2x … 2.5x**

(depends on matrix size, parallel scale, number of RHS)

Motivation

- No available universal CAMG implementation for multiple RHS

- New C++11 code base allows experiments with up-to-date ideas of improving sparse solvers

XAMG architecture highlights

- Use *hypre* library code for CAMG hierarchy construction

«HYPRE: High performance preconditioners»
`http://www.llnl.gov/CASC/hypre/`

- We do not extend or fork *hypre* code, just use it for hierarchy construction

- Special mode: per-level hierarchy

XAMG architecture highlights

- Number of RHS as a template parameter
- Sets up the number of vectors at compile time
- So, compiler is able to generate vector instructions for inner loops
- Index and value types (integer and floating point) are also template parameters

XAMG architecture highlights

- Variative choice of sparse matrix storage format
- Matrix is dynamically polymorphic: inheritance
- It is possible to combine different storage formats to get best productivity

  → for different multigrid hierarchy levels
  → for parts of a single matrix

XAMG architecture highlights

- Index compression: detect which integer index data size is enough for each hierarchy level

- Floating point size: 32-bit floating point precision instead of 64-bit for smaller hierarchy levels

- Combined dynamic and static polymorphism: «creator» functions for matrix objects are huge (large **`if-else`** trees)
  → automatic code generation is used:
  `https://github.com/a-v-medvedev/cppcgen`

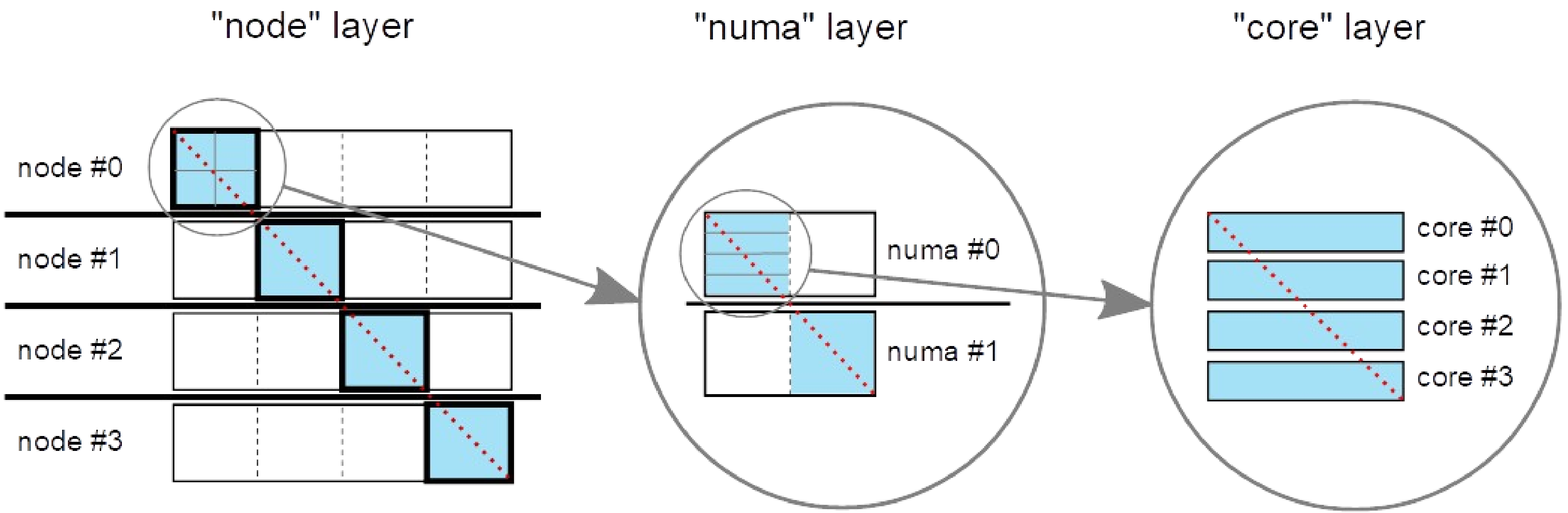
XAMG architecture highlights

- **MPI+ShM** hybrid parallel programming model

- On communication level:
  → decomposition of parallel communications into intra-node
  and inter-node levels
  → implementation of intra-node communications using communication
  POSIX shared memory primitives

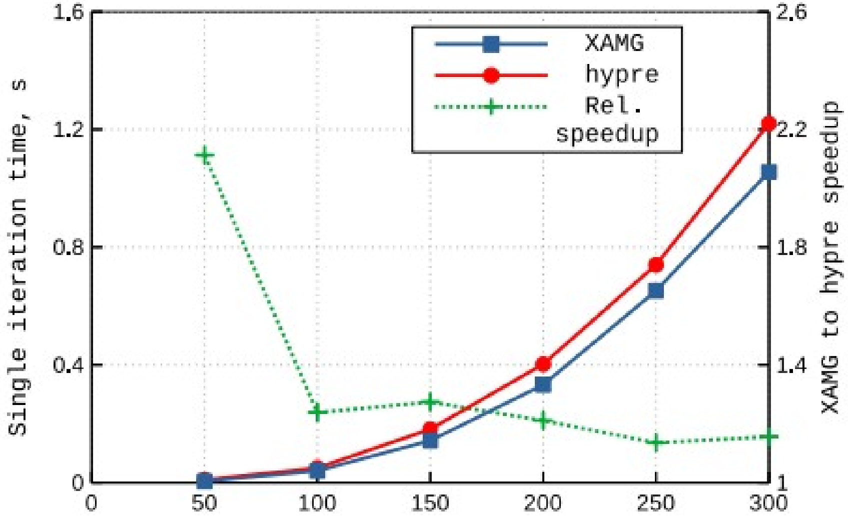
- On data level:
  → matrices and vectors are allocated in POSIX shared
  memory and split specifically

MPI+ShM

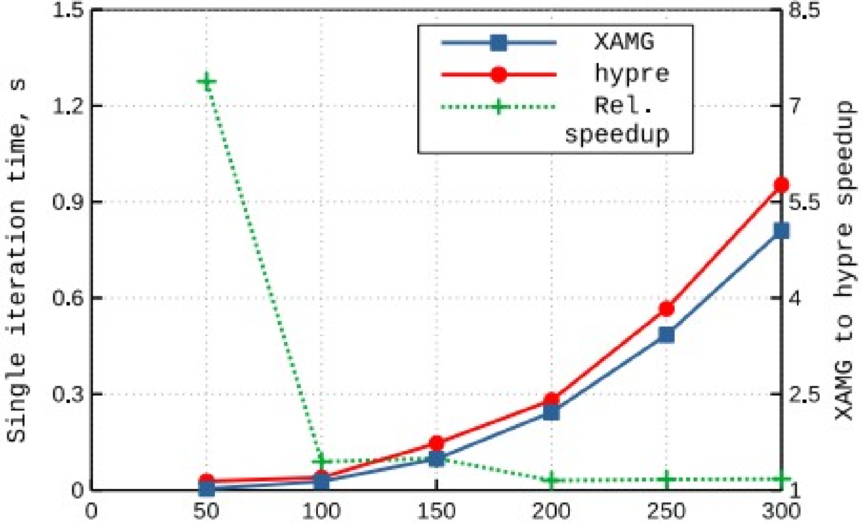

"node" layer
"numa" layer
"core" layer
node #0
node #1
node #2
node #3
numa #0
numa #1
core #0
core #1
core #2
core #3

Performance: XAMG vs. *hypre*

Lomonosov-2



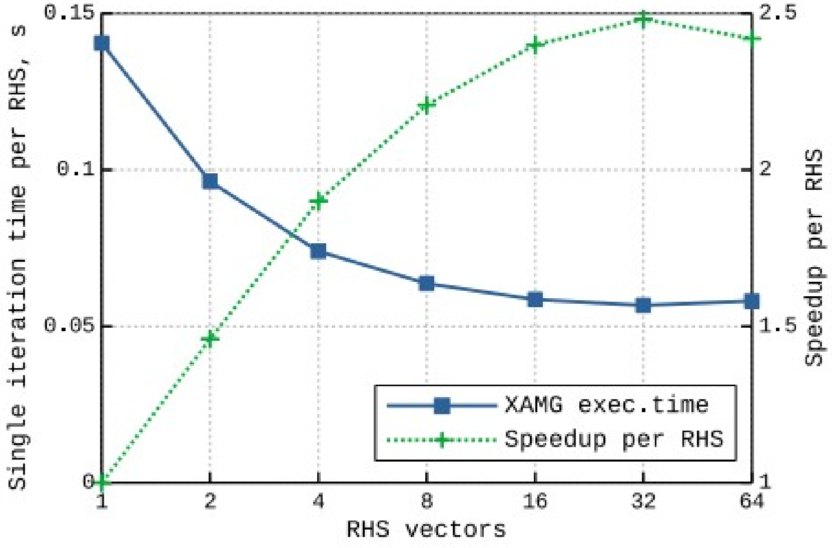
HPC4



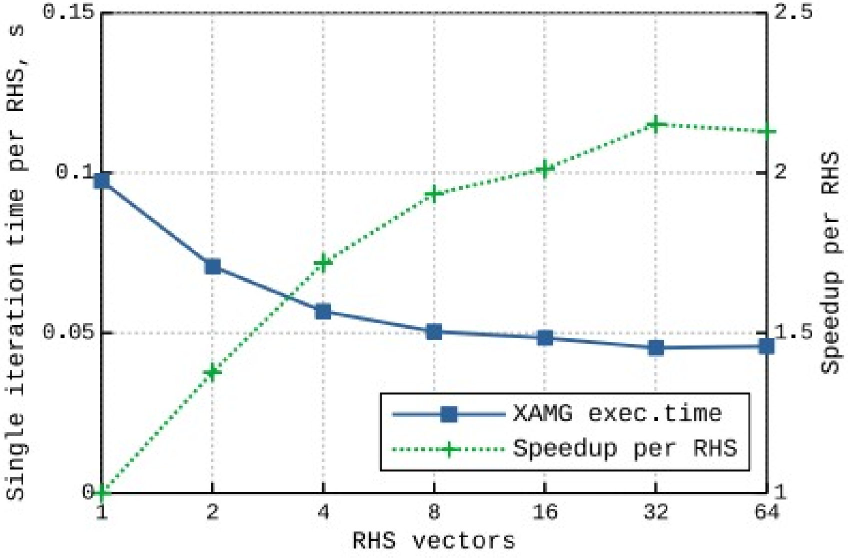
Single node; Poisson cubic grid (size = 50^3...300^3); Pure MPI mode

Performance: multiple RHS

Lomonosov-2



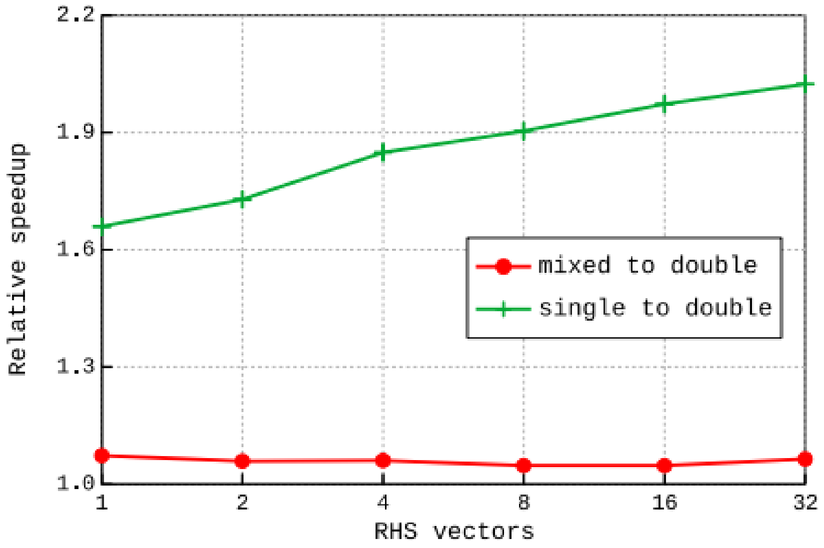
HPC4



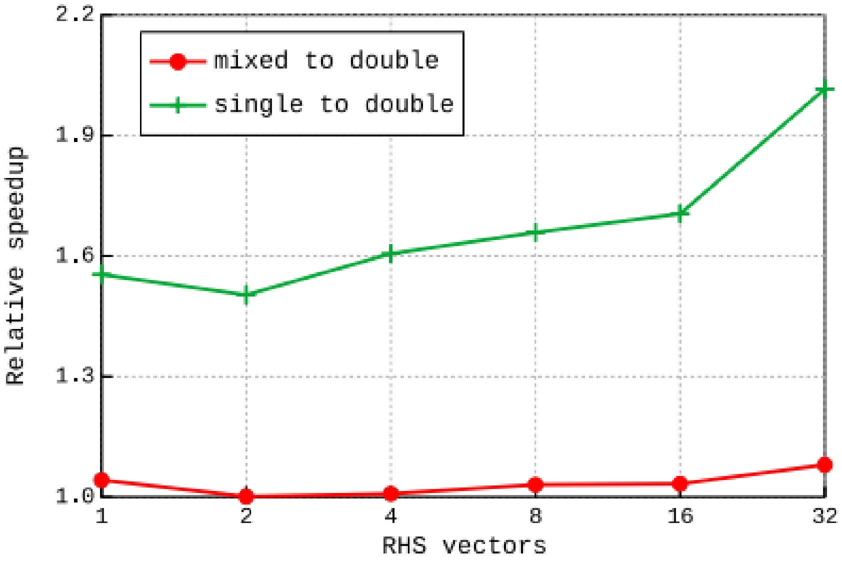
Single node; Poisson cubic 150^3; Pure MPI mode

Performance: mixed precision

Lomonosov-2

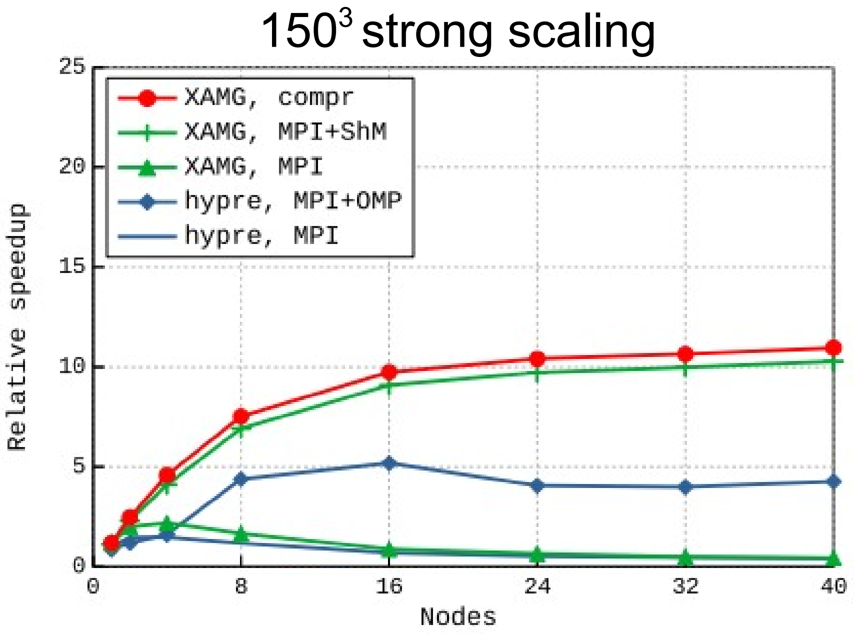


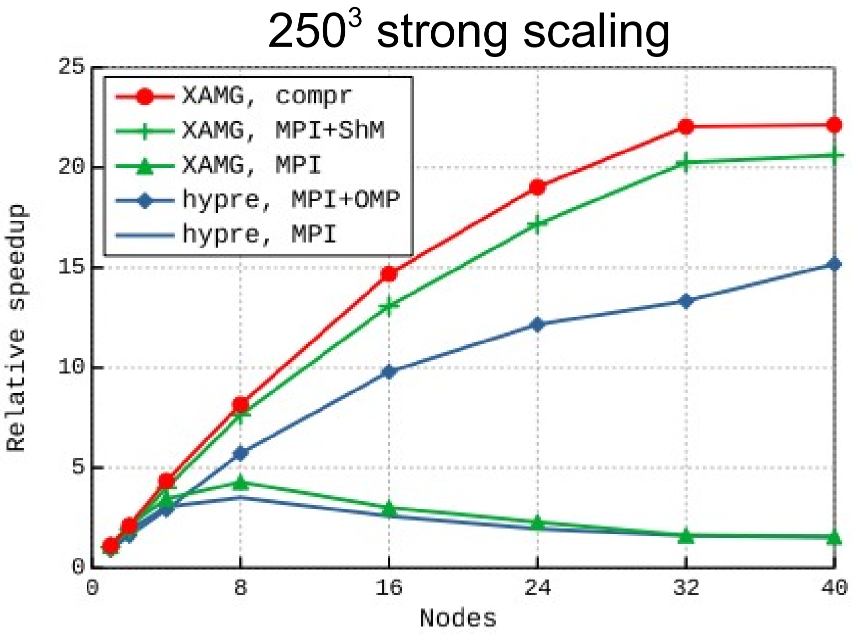
HPC4



Single node; Poisson cubic 200^3; Pure MPI mode

Performance: scalability

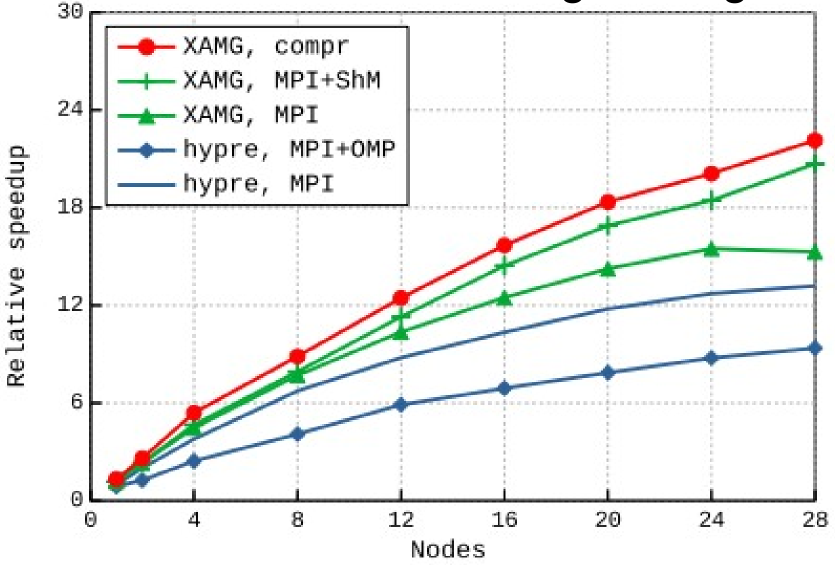




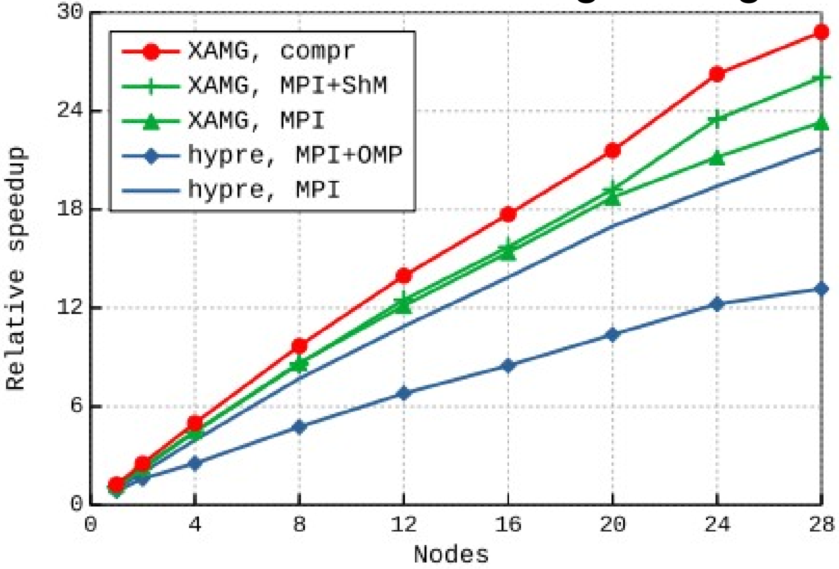
HPC4, 2..40 nodes; Poisson cubic 150^3; and 250^3;

Performance: scalability

2.7 mln unkn. strong scaling



9.7 mln unkn. strong scaling



Lomonosov-2, 2..28 nodes; Poisson channel flow problem

Conclusions

- Multiple RHS feature improves calculation times
- Complex code architecture opens up the way to implement:
  - automatic index data type compression
  - mixed precision of floating point data
  - combination of different matrix storage formats
  - complex algorithms like Iterative Refinement
  - advanced per-level CAMG configuration & auto-tuning
- Index and data compression improves productivity and scalability
- MPI+ShM parallel programming model improves scalability significantly

Future work

- GPU solver implementation (*WIP*)
- More advanced matrix storage formats and their combination (*WIP*)
- FP16 for mixed precision
- Automatic optimization of per-level CAMG tuning parameters (*WIP*)
- Connection with real-world applications

Thank you!

e-mail: krasnopolsky@imec.msu.ru
e-mail: a.medvedev@imec.msu.ru

НАУЧНАЯ КОНФЕРЕНЦИЯ
Суперкомпьютерные дни в России

The presented work is supported by RSF grant No. 18-71-10075.
The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

Hardware

- Lomonosov-2:

Intel Xeon E5-2697v3; 1xCPU: 14 cores
Infiniband FDR

- HPC4:

Intel Xeon E5-2680v3; 2xCPU: 24 cores
Infiniband QDR